REMARKS

Claims 1, 7, 14, 20, and 32 have been amended for clarification only. No claims have been added or cancelled. Therefore, claims 1-35 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 102(b) Rejection:

The Examiner rejected claims 1, 2, 11-15, 24-26 and 32-34 under 35 U.S.C. § 102(b) as being anticipated by Rotenberg, et al. (hereinafter "Rotenberg"). Applicants traverse this rejection for at least the following reasons.

Regarding claim 1, the Examiner asserts that Rotenberg teaches a prefetch unit coupled to the instruction cache, the branch prediction unit, and the trace cache in FIG. 4 and Section 2.2. The Examiner submits that Rotenberg's instruction latch appears to fill the role of Applicants' *prefetch unit*. However, Applicants assert that one of ordinary skill in the art could not reasonably consider a simple latch, as described in Rotenberg, to be the *prefetch unit* of the present invention, and this latch is not described as having any of the functionality of the *prefetch unit* recited in Applicants' claims.

For example, claim 1 recites, "if the prefetch unit identifies a match for the predicted target address in the trace cache..." Rotenberg's instruction latch clearly does not identify a match for a predicted target address in the trace cache, as it simply serves to latch the output of a 2:1 mux and does not even determine which of the inputs of the mux to latch (see FIG. 4). There is clearly nothing in Rotenberg that discloses this latch performing a comparison of a predicted target address to addresses in the trace cache in response to a branch prediction unit outputting the predicted target address. Instead, logic within Rotenberg's trace cache itself (hit logic, FIG. 4) determines if there is a hit for a branch prediction and if not, an instruction from the instruction cache is latched.

In addition, claim 1 recites, *the prefetch unit is configured to fetch instructions from the instruction cache until the branch prediction unit outputs a predicted target address; wherein the prefetch unit is configured to check the trace cache for a match for the predicted target address in response to the branch prediction unit outputting a predicted target address; wherein the prefetch unit is configured to not check the trace cache for a match until the branch prediction unit outputs a predicted target address; and wherein if the prefetch unit identifies a match for the predicted target address in the trace cache, the prefetch unit is configured to fetch one or more of the plurality of traces from the trace cache.* Contrary to the Examiner's assertion, Rotenberg's instruction latch clearly does not perform these functions. Nor does any component or combination of components in Rotenberg perform these functions. The Examiner submits that Rotenberg teaches these limitations in Section 2.2, "<u>on a trace cache miss</u>, fetching proceeds normally from the instruction cache" and "on a trace cache hit, an entire trace of instructions is <u>fed into the instruction latch</u>, bypassing the instruction cache." However, as illustrated in FIG. 4, Rotenberg's <u>core fetch unit</u>, not the instruction latch, fetches instructions from the instruction cache (see, e.g., Section 2.1, paragraph 1-3) and outputs them to a 2:1 mux. In addition, Section 2.2, paragraph 3 states, "The trace cache is <u>accessed in parallel with</u> the instruction cache and BTB using the current fetch address." Thus, contrary to the limitations recited in Applicants' claim 1, instructions are always fetched from the instruction cache <u>and</u> the trace cache in parallel, and these two sets of instructions are output to the mux illustrated in FIG. 4. There is nothing in Rotenberg that discloses a <u>prefetch unit</u> that <u>fetches instructions from an instruction cache until a branch prediction unit outputs a predicted target address</u>, that <u>only then checks the trace cache for a match</u>, and that fetches instructions from the trace cache if a match is found. The <u>fetching</u> operation is not performed by Rotenberg's instruction latch, as the Examiner suggests, nor does it change when a predicted target address is output, as recited in Applicants' claim 1. Instead, the determination of whether there is a trace cache hit (performed in the trace cache itself) selects which inputs of the mux (instructions already fetched from the instruction cache or instructions <u>already fetched from the trace cache as a result of a search for an address match</u>) are fed to the instruction latch. In contrast to the limitations of Applicants' claim 1, the search for a match in

Rotenberg's trace cache is always done, regardless of whether or not a branch predication unit outputs a predicted target address.

Finally, fetching from the instruction cache in response to detecting a trace cache miss and fetching from the trace cache in response to detecting a trace cache hit are clearly not the same as searching the trace cache only in response to outputting a predicted target address, i.e., one for which a comparison to addresses in the trace cache has not yet taken place. There is clearly nothing in Rotenberg to teach or suggest that a prefetch unit only checks the trace cache for a match *in response to the branch prediction unit outputting a predicted target address*, and that it does not search the trace cache for a match *until the branch prediction unit outputs a predicted target address*, as recited in Applicants' claim 1.

In the Response to Arguments section of the Final Office Action, the Examiner states that he has interpreted "fetching an instruction" to be the act of providing an instruction to the computer system. Using that definition, the Examiner states that the instruction latch appeared to perform the role of the prefetch unit in Applicants' claims. To clarify his position, the Examiner submits, "the instruction latch fulfills the role of the prefetch unit, when its inputs are considered as well, which is why the rejection brought in what the trace cache and the core fetch unit was doing, as they both fed into the latch, which then finished the fetch operation by providing the appropriate instruction to the system, through either the trace cache or the core fetch unit. Given this clarification, Examiner believes the issues involving finding a match in the trace cache, and fetching instructions from an instruction cache will be resolved."

Applicants assert, however, that claim 1 does not recite finding a match in the trace cache and fetching from an instruction cache if a match is not found, as the Examiner describes. In contrast, claim 1 requires that the trace cache not be searched for a match until the branch prediction unit outputs a predicted target address and if there is a match, fetching instructions from the trace cache. Claim 1 does not recite what happens if a match is not found in the trace cache. Furthermore, the combination of Rotenberg's

instruction latch and inputs still do not teach the limitations of claim 1 discussed above, i.e., that the trace cache is not checked for a match until the branch prediction unit outputs a predicted target address.

The Examiner further submits, in the Response to Arguments section, that the limitation of "fetches from the instruction cache until the branch prediction unit outputs a predicted target address" had been interpreted to read as "when the branch prediction unit outputs a predicted target address *not* in the trace cache, as otherwise, the invention would clearly not function as described in the Applicants arguments. Applicant has argued that in the claimed invention, the prefetch unit would stop fetching from the instruction cache as a result of a predicted target address being generated, which would cause the processor to stop running if the value was not in the trace cache, which Examiner interpreted would not be the case, as that would not be an enabled system, rather, Examiner read the last two limitations together, in which the instruction cache is no longer output to the system when there is a trace cache hit from the predicted address, but if there is a predicted address NOT in the trace cache, the processor would continue to run instead of shutting down."

Applicants assert that the Examiner has misinterpreted Applicants' argument. Applicants' argument was that claim 1 requires that the prefetch unit fetches from the instruction cache until a predicted target address is output by the branch prediction unit, and that a search for a match in the trace cache (and subsequently, a fetch from the trace cache) does not take place until the branch prediction unit outputs a predicted target address. That is, the prefetch unit will not check for the match if the branch prediction unit does not output a predicated target address. Claim 1 has nothing to do with what happens if the match is not found, as the Examiner suggests. The Examiner is correct that, according to claim 1, if the branch predication unit outputs a predicted target address and there is a match for the predicted target address in the trace cache, the prefetch unit will fetch from the trace cache. Using the Examiner's interpretation of "fetching", this is also true in Rotenberg (i.e., if there is a match in the trace cache, an instruction will be provided from the trace cache). However, as discussed above, Rotenberg's trace cache

operation is clearly different from Applicants' claimed invention, in that it teaches searching for a hit in the trace cache first, and then fetching from the instruction cache if there is not a hit.

For the above reasons, the original claim 1 distinguished over Rotenberg. However, claim 1 has been amended to clarify the meaning of the claim. For at least the reasons above, Rotenberg does not teach many of the limitations of claim 1 and cannot be said to anticipate claim 1. Applicants again remind the Examiner that anticipation requires the presence in a single prior art reference disclosure of <u>each and every limitation</u> of the claimed invention, <u>arranged as in the claim</u>. M.P.E.P 2131; *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The **<u>identical</u>** invention must be shown <u>in as complete detail</u> as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). For at least the reasons discussed above, Rotenberg clearly cannot be said to anticipate claim 1. Thus, the rejection of claim 1 is not supported by the cited art and removal thereof is respectfully requested.

Claim 14 includes limitations similar to claim 1, and so the arguments presented above apply with equal force to this claim, as well.

Regarding claim 32, contrary to the Examiner's assertion, Rotenberg fails to teach or suggest *continuing to fetch instructions from the instruction cache <u>without searching a trace cache</u> until a branch target address is generated.* Again, the Examiner cites Section 2.2 "on a trace cache miss, fetching proceeds normally from the instruction cache" as teaching this limitation. However, as discussed above, there is nothing in Rotenberg that discloses fetching instructions from the instruction cache <u>without searching a trace cache</u> until a branch target address is generated. Instead, FIG. 4 and its description clearly illustrate that instructions are continuously fetched from the instruction cache by the core fetch unit and that instructions are checked for a match and fetched from the trace cache in parallel, although the instructions fetched from the instruction cache may not be latched and passed to the decoder if the alternate input

(from the trace cache) is selected at the 2:1 mux. Furthermore, detecting a trace cache miss is clearly not the same as generating a branch target address, one for which a comparison to addresses in the trace cache has not yet taken place. There is nothing in Rotenberg to teach or suggest fetching from the instruction cache until a branch target address is generated, as recited in claim 32. In fact, claim 32 further recites, "if a branch target address is generated, searching a trace cache for an entry corresponding to the branch target address." Therefore, in the claimed invention, the trace cache is not searched for a cache hit unless a branch target address is output, and then if there is a match, the prefetch unit stops fetching instructions from the instruction cache. In Rotenberg, by contrast, the trace cache is accessed in parallel with the instruction cache and BTB (branch target buffer) using the current fetch address (see, e.g., Section 2.2, paragraph 3). There is nothing in Rotenberg that teaches or suggests that the trace cache is searched only if a branch target address is generated, as in Applicants' claim, and this paragraph teaches away from this limitation.

In the Response to Arguments section of the Final Office Action, the Examiner submits, "Firstly, as explained above, while the two are accessed in parallel, only one can be "fetched", that is, sent to the system, therefore, if the trace cache generates a hit (does not generate a miss), the instruction cache, while being accessed, will not send instructions to the system, effectively not fetching." Applicants note, however, that claim 32 does not recite fetching instructions from the trace cache on a trace cache hit and fetching instructions from the instruction cache on a trace cache miss, as the Examiner suggests. Instead, claim 32 recites *continuing to fetch instructions from the instruction cache without searching a trace cache until a branch target address is generated*; and *in response to a branch target address being generated, searching a trace cache for an entry corresponding to the branch target address*.

The Examiner further submits, in the Response to Arguments section, that he interpreted "until a branch target address is generated" to read on the case the trace cache was a miss as well, to avoid an enablement issue of the processor shutting down, unable to fetch instructions from either source. This interpretation is clearly incorrect, since this

claim has nothing to do with the results of the search for an entry corresponding to the branch target address. Claim 32 includes the limitation that the trace cache is not searched for a matching entry until a branch target address is generated, which is not taught by Rotenberg. Claim 32 does not recite anything about the results being a hit or miss, or what happens in response to a hit or miss. Therefore, **claim 32 clearly does not require that there must have been a trace cache miss in order to fetch instructions from the instruction cache,** as suggested by the Examiner, since the trace cache is not checked for a hit or miss for every instruction. Instead, claim 32 recites that after fetching from the instruction cache, the prefetch continues to fetch from the instruction cache without searching the trace cache until a branch target address is generated. Only then is the trace cache searched for a match.

For at least the reasons above, Rotenberg clearly cannot be said to anticipate claim 32. Therefore, the removal of the rejection of claim 32 is respectfully requested.

Section 103(a) Rejections:

The Examiner rejected claims 3 and 16 under 35 U.S.C. § 103(a) as being unpatentable over Patterson, et al. (hereinafter "Patterson"), claims 4, 10, 17 and 23 as being unpatentable over Rotenberg in view of Braught, claims 5-8 and 18-21 as being unpatentable over Rotenberg and Braught, further in view of Lange et al. (U.S. Patent 3,896,419) (hereinafter "Lange"), claims 9 and 22 as being unpatentable over Rotenberg in view of Akkary et al. (U.S. Patent 6,247,121) (hereinafter "Akkary"), and claims 27-31 and 35 as being unpatentable over Rotenberg, Braught, Lange in view of Akkary. Applicants traverse this rejection for at least the following reasons.

Regarding claim 27, the Examiner submits that Rotenberg teaches *a method... comprising starting construction of a new trace if the received instruction is associated with a branch label* (Section 2.2, Rotenberg starts traces on branches, which branch to labels – or addresses, as shown by Braught), but fails to teach *receiving a retired*

instruction. The Examiner admits that Rotenberg does not teach that instructions need to be retired before the trace can be generated and relies on Akkary to teach this limitation. The Examiner submits that Akkary teaches that instructions are not put into the trace buffers until they have been retired, to ensure that they executed correctly (column 3, lines 40-44). This passage includes the following description:

> Final retirement logic 134 finally retires instructions in trace buffers 114 after it is assured that the instructions were correctly executed either originally or in re-execution.

The Examiner has incorrectly interpreted this passage as teaching that instructions are not put into the trace buffers until they have been retired. **This passage actually says just the opposite**, that is, that instructions placed in the trace buffer stay in the trace buffer until they are retired. This is clear from the following two passages (emphasis added):

> column 6, lines 53-56:
>
> all needed details regarding the instruction are maintained in trace buffers 114 and MOB 178 until a final retirement, described below.
>
> column 10, lines 1-10:
>
> In one embodiment of the invention, final retirement includes… (3) deallocation of trace buffer and MOB 178 resource entries.

Therefore, Rotenberg and Braught in view of Akkary clearly fails to teach or suggest this limitation of Applicants' claim 27.

The Examiner also admits that Rotenberg fails to teach *if a previous trace under construction duplicates a trace in a trace cache, delaying construction of the new trace until the received instruction corresponds to a branch label*. The Examiner submits that Lange teaches a system in which a cache is checked for a match with memory, while the memory is being read at the same time, with the advantage that there is no delay in the overall data fetch cycle if there is no match in the cache, and the memory access can be cancelled before it incurs any system slowdown as well (column 2, lines 13-31, and column 5, lines 5-10). Given this advantage, the Examiner asserts that it would have

been obvious to one of ordinary skill in the art at the time the invention was made to check the trace cache for a duplicate copy of the trace at the same time the trace was being constructed, in order to minimize or eliminate delay in generating the trace in the case that the trace is not in the cache, and aborting the unnecessary trace if it is found in the cache. The Examiner further asserts that when a trace is found to be in the cache, and the trace generation is aborted, it is obvious that a new trace would not be built until a new branch (which would have to be associated with a branch label) was retired. **The Examiner's reasoning is clearly flawed.** First, claim 27 does not recite fetching data from memory or with checking for a cache hit on a memory access. Instead, this claim describes construction of a trace from retired (e.g., completed) instructions. The construction of a trace from retired instructions does not require fetching data from memory and, thus, the issue suggested by the Examiner (delay in generating the trace in the case that the trace is not in the cache) would not exist. The Examiner seems to suggest that generating a trace in a trace cache necessarily results in a system slowdown, providing motivation to avoid construction of duplicate traces. However, the Examiner has not provided any evidence that this is the case.

In the Response to Arguments section of the Final Office Action, the Examiner submits, "While Applicant and Examiner are in agreement that instructions are retired from the trace buffers on retirement, the quoted passage of Akkary was intended by the Examiner to show that instructions cannot be shown to have executed correctly until they had been retired, and was not used to show how Akkary's trace buffers functioned. Thus, the Examiner stated that the motivation to combine Akkary with Rotenberg was that one of ordinary skill in the art would recognize the advantage of waiting to add an instruction to a trace until the instruction was guaranteed to have executed correctly, to avoid incorrect traces, as traces are used to increase performance, not degrade it by giving incorrect results. Therefore, Akkary suggests the limitation of using a retired instruction, by providing one of ordinary skill in the art with a motivation for using a retired instruction, as opposed to in-flight instructions."

First, Applicants assert that Akkary does not suggest "using a retired instruction" for any particular purpose, as the Examiner suggest, but only teaches that an instruction should not be retired until it has executed correctly either originally or when re-executed, such as during recovery from a misspeculation. This has nothing to do with building "incorrect traces", which the Examiner suggests degrade performance by "giving incorrect results." The Examiner seems to be implying that in the system of Rotenberg, in which traces are built from unretired instructions, executing certain of these traces (i.e., "incorrect traces") can lead to incorrect results. This logic is clearly flawed. First, it is not clear what the Examiner means by "incorrect traces." The trace logic in Rotenberg is not used to supply <u>results</u> of a previously executed trace, but to provide a series of <u>instructions</u> that may have been executed before. In addition, the system of Rotenberg may store traces of instructions for multiple possible paths, such as if a predicated taken path is stored for a given trace and the path actually taken is stored later (after a trace cache miss). The trace cache may therefore provide different traces for paths that were predicted to be taken or not taken, but the fact that the chosen path may or may not have been <u>retired</u> when previously executed (or when stored without being executed) has nothing to do with whether it may give <u>incorrect results</u> the next time it is executed, e.g., due to being mispredicted. Therefore, the Examiner's suggested reasons for combining Akkary with Rotenberg are clearly not supported by the cited art.

The Examiner further submits, in the Response to Arguments section, "Examiner and Applicant are in complete agreement that Claim 27 does not recite fetching data from memory, or checking for a cache hit on a memory access. Lange was, however, not used to teach limitations such as those. Instead, Lange was used to teach why it would be advantageous to check for a duplicate copy of a value in a cache, and why you would not want to continue with the operation that is not needed when the value is already present, in Lange's case, it was to prevent unnecessary memory access that could slow down the system. There is a performance degradation in Rotenberg's system that does come up in this case, as seen in Section 2.3, Point 5. Here, Rotenberg discusses what happens when a trace cache miss occurs when the line-fill buffer is collecting a trace. If this line-fill buffer was collecting a trace which was already in the buffer, then a performance hit

would occur, as the options are to ignore it, which would remove the performance boost of the cache, to delay servicing it, or to provide multiple buffers, which would create the need for more hardware, which could increase the cost of the system, and also raises the complexity. Given the less complex methods, and that they both involve lowering the efficiency of the processor, being able to keep the line-fill buffer free of unnecessary data clearly provides an increase in performance, and is a problem which Rotenberg has disclosed which Lange provides a solution for, by teaching the idea that a check for duplicate data can end an unnecessary operation."

The Examiner's logic is clearly flawed. First, in the system of Rotenberg, the fill-line buffer logic services <u>trace cache misses</u> (Section 2.2, fourth paragraph). That is, the fill-line buffer fetches instructions because the combination of a matching target address and matching branch flags <u>is not found in the trace cache</u>. Therefore, there is no reason to believe there would ever be a duplicate trace in the system of Rotenberg. The performance problem described in Section 2.3, point 5, therefore, would not be solved by checking for duplicate data before building a trace. Since duplicate traces should not be contained in the trace cache of Rotenberg, there would be no opportunity to avoid such duplication.

Second, Applicants assert that the Examiner's general statements about "lowering the efficiency of the processor" and "teaching the idea that a check for duplicate data can end an unnecessary operation" are broad, conclusory statements that teach nothing about the specific limitations of Applicants' claim. First, the "performance problem" described in Rotenberg in Section 2.3, point 5 (the fill-line buffer may be busy collecting a new trace when access is requested) is not analogous to the performance problem solved by Lange (fetching data from main memory is slower than fetching data from a cache.) Therefore, one of ordinary skill would have no motivation to apply the solution described in Lange (i.e., check the cache before fetching from main memory) to the problem of Rotenberg. In addition, a method for reducing fetches from memory by first checking a cache is not analogous to a method for reducing the amount of data that needs to be loaded into a trace cache by checking the contents of the trace cache itself. Finally, even

if the problems were analogous, checking Rotenberg's cache before collecting a new trace would not solve the problem described in Section 2.3, point 5, since there should be no duplicate traces generated in Rotenberg and, therefore, no opportunity to "end an unnecessary operation," as suggested by the Examiner. Therefore, the Examiner's reasons for combining the references are clearly not supported by the cited art.

Applicants remind the Examiner that to establish a *prima facie* obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. Obviousness cannot be established by combining or modifying the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion or incentive to do so. *In re Bond*, 910 F. 2d 81, 834, 15 USPQ2d 1566, 1568 (Fed. Cir. 1990). As discussed above, the cited references do not teach or suggest all limitations of claim 27, nor do they include any suggestion or incentive to modify their teachings to produce the claimed invention.

For at least the reasons above, the rejection of claim 27 is not supported by the cited art and removal thereof is respectfully requested.

Claim 35 includes limitations similar to claim 27, and so the arguments presented above apply with equal force to this claim, as well.

Regarding claim 5, contrary to the Examiner's assertion, Rotenberg in view of Braught and Lange fails to teach or suggest *wherein the trace generator is configured to check the trace cache for a duplicate copy of the trace that the trace generator is constructing*. The Examiner admits that Rotenberg in view of Braught fails to teach checking the trace cache for a duplicate copy of the trace that is being constructed, and relies on Lange to teach this limitation using arguments similar to those presented regarding claim 27. However, as discussed above, the problem solved by Lange is not at all analogous to any disclosed problems in Rotenberg, nor would the combination of

Lange, Rotenberg, and Braught solve any problems disclosed in Rotenberg or teach this limitation of Applicants' claimed invention.

For at least the reasons above, the rejection of claim 5 is not supported by the cited art and removal thereof is respectfully requested.

Claims 18 and 29 include limitations similar to claim 5, and so the arguments presented above apply with equal force to these claims, as well.

Regarding claim 6, contrary to the Examiner's assertion, Rotenberg in view of Braught and Lange fails to teach or suggest *wherein if the trace generator identifies a duplicate copy of the trace, the trace generator is configured to discard the trace under construction*. The Examiner cites Lange, column 5, lines 5-10 as teaching this limitation. However, this passage describes blocking retrieval of data from memory if it is first found in the cache. **This teaches nothing about discarding a trace under construction (for which an instruction has already been fetched from memory.)** Therefore, the combination of Rotenberg, Braught, and Lange does not teach this limitation, nor is there motivation to combine the references, as discussed above.

For at least the reasons above, the rejection of claim 6 is not supported by the cited art and removal thereof is respectfully requested.

Claims 19 and 31 include limitations similar to claim 6, and so the arguments presented above apply with equal force to these claims, as well.

Regarding claim 7, contrary to the Examiner's assertion, Rotenberg in view of Braught and Lange fails to teach or suggest *wherein in response to the trace generator identifying an entry corresponding to a duplicate copy of the trace, the trace generator is configured to check the trace cache for an entry corresponding to a next trace to be generated*. The Examiner submits that Rotenberg teaches the trace caches is checked every time there is a potential new trace, so when one trace is found and discarded, the

next potential new trace will cause the trace cache to be checked again. First, as discussed above, Rotenberg in view of Braught and Lange does not teach checking the trace cache for duplicate traces at all. Furthermore, claim 7 does not recite merely checking the next trace that is generated (i.e., checking every trace as it is about to be generated), but instead requires <u>checking the trace cache for an entry corresponding to the next trace to be generated</u> (i.e., not the trace currently being generated) <u>in response to identifying an entry corresponding to a duplicate copy of the trace</u> (i.e., the trace currently being generated). Rotenberg in view of Braught and Lange clearly does not teach this limitation.

For at least the reasons above, the rejection of claim 7 is not supported by the cited art and removal thereof is respectfully requested.

Claim 20 includes limitations similar to claim 7, and so the arguments presented above apply with equal force to this claim, as well.

Regarding claim 8, contrary to the Examiner's assertion, Rotenberg in view of Braught and Lange fails to teach or suggest *wherein if the trace generator identifies a trace entry corresponding to the next trace to be generated, the trace generator is configured to discard the trace under construction*. The Examiner again submits that Lange teaches this limitation in Column 5, lines 5-10. However, as discussed above regarding claim 6, this passage describes blocking retrieval of data from memory if it is first found in the cache. **This teaches nothing about discarding a trace under construction (for which an instruction has already been fetched from memory), much less discarding a trace under construction if a trace cache includes an entry corresponding to <u>the next trace to be constructed</u> (i.e., not the trace under construction.)** Therefore, the combination of Rotenberg, Braught, and Lange clearly does not teach this limitation, nor is there motivation to combine the references, as discussed above.

For at least the reasons above, the rejection of claim 8 is not supported by the cited art and removal thereof is respectfully requested.

Claim 21 includes limitations similar to claim 8, and so the arguments presented above apply with equal force to this claim, as well.

Regarding claim 9, contrary to the Examiner's assertion, Rotenberg in view of Akkary fails to teach or suggest *wherein the trace generator is configured to generate traces in response to instructions being retired.* The Examiner admits that Rotenberg does not teach that instructions need to be retired before the trace can be generated and relies on Akkary to teach this limitation. However, as discussed above regarding claim 27, the Examiner's reason for combining the references is not supported by the cited art. In addition, claim 9 does not recite that it is necessary to retire instructions before generating traces, as the Examiner suggests. Instead, it recites that *the trace generator is configured to generate traces* <u>*in response to*</u> *instructions being retired.* Applicants assert that even if Rotenberg in view of Akkary taught that instructions need to be retired before a trace may be generated, this does not teach that the trace generator is configured to generate traces <u>in response to</u> instructions being retired, as recited in claim 9. Being a <u>necessary condition</u> is not the same as initiating <u>a response</u>.

For at least the reasons above, the rejection of claim 9 is not supported by the cited art and removal thereof is respectfully requested.

Claim 22 includes limitations similar to claim 9, and so the arguments presented above apply with equal force to this claim, as well.

Regarding claim 10, contrary to the Examiner's assertion, Rotenberg in view of Braught fails to teach or suggest *wherein the trace generator is configured to generate traces in response to instructions being decoded.* The Examiner submits that Rotenberg teaches the trace cannot be completed (written into the cache) until the trace target addresses are calculated, which requires the instructions to be decoded. Applicants

assert, however, that even if Rotenberg requires instructions to be decoded before they may be written into the cache, this does not teach that the trace generator is configured to generate traces in response to instructions being decoded, as recited in claim 10. Being a necessary condition is not the same as initiating a response.

For at least the reasons above, the rejection of claim 10 is not supported by the cited art and removal thereof is respectfully requested.

Claim 23 includes limitations similar to claim 10, and so the arguments presented above apply with equal force to this claim, as well.

Applicants also assert that the rejection of numerous other ones of the dependent claims is further unsupported by the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and prompt notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5500-88700/RCK.

Also enclosed herewith are the following items:

- ☐ Return Receipt Postcard
- ☐ Petition for Extension of Time
- ☐ Notice of Change of Address
- ☐ Other:

Respectfully submitted,

Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: October 23, 2006